

# HEISENBERG GROUPS AS PLATFORM FOR THE AAG KEY-EXCHANGE PROTOCOL

DELARAM KAHROBAEI, HA T. LAM

**ABSTRACT.** Garber, Kahrobaei, and Lam studied polycyclic groups generated by number field as platform for the AAG key-exchange protocol. In this paper, we discuss the use of a different kind of polycyclic groups, Heisenberg groups, as a platform group for AAG by submitting Heisenberg groups to one of AAG's major attacks, the length-based attack.

After the introduction of the Anshel-Anshel-Goldfeld (AAG) key-exchange protocol in 1999 [1], it has been studied extensively with different groups as platform, for example, using braid groups by Ko et al. [17], using Thompson's group by Shpilrain and Ushakov [23]. Different attack methods have also been applied to AAG [18, 12, 21].

One of the major attack methods of the AAG protocol is the length-based attack (LBA), originated with Hughes and Tannenbaum [14], whose paper provided an example in braid groups, together with remark on the importance of the length function. Garber et al. studied the infeasibility of the length based attack with a choice of length function [11], but then introduced a variant of it using memory which succeeded in breaking AAG for braid group [10]. Myasnikov and Ushakov also studied the length-based attack for braid group and provided several variants with which it was possible to break AAG [20]. Similar attack was implemented against system based on the Thompson group [22]

In 2004, Eick and Kahrobaei suggested using polycyclic groups as platform group for the AAG key-exchange protocol [4]. This idea was realized by Garber, Kahrobaei, and Lam [9]. In that paper, several variants of LBA were tested on an AAG implementation using polycyclic groups generated from number field. The result suggests that this type of polycyclic group is resistant to the length-based attack. Taking inspiration from this, we want to study the Heisenberg group as platform group for AAG. We use the variants of the length-based attack presented in [9] to conduct tests on an implementation of the Heisenberg group. The result is then analysed, and we conclude that the Heisenberg groups can be used as platform for the AAG protocol given the correct parameters.

Furthermore, the conjugacy search problem has been used for several other cryptographic protocols, such as the non-commutative Diffie-Hellman key exchange [17], the non-commutative El-Gamal key exchange [15], the non-abelian Cramer-Shoup key exchange [2] and the non-commutative digital signatures [16]. The length-based attack can be applied to all of these protocols, hence testing different groups against it and collecting data about parameters that make them resistant to LBA is important, not just for the implementation of AAG but also of other protocols.

The paper is organized as follows: in Section 1, we introduce the Anshel-Anshel-Goldfeld key exchange protocol, in Section 2, there is a short review of polycyclic

groups and the construction of Heisenberg group; in Section 3, we talk about the length-based attack; and in Section 4, we detail the experiments, results and conclusion that we made.

## 1. THE ANSHEL-ANSHEL-GOLDFELD KEY EXCHANGE PROTOCOL AND HEISENBERG GROUP

In this section, we give a short introduction to the Anshel-Anshel-Goldfeld key-exchange protocol. As usual, we use Alice and Bob as two parties who want to communicate over an insecure channel. The Anshel-Anshel-Goldfeld key exchange protocol works as follows:

Let  $G = \langle g_1, \dots, g_n \mid R \rangle$  be a finitely presented group with generators  $g_1, \dots, g_n$  and relation set  $R$ . First, Alice chooses, as her public set,  $\bar{a} = (a_1, \dots, a_{N_1})$  where  $a_i \in G$  and Bob chooses, as his public set,  $\bar{b} = (b_1, \dots, b_{N_2})$  where  $b_i \in G$ . They both publish their sets. Alice then chooses her private key  $A = a_{s_1}^{\varepsilon_1} \dots a_{s_L}^{\varepsilon_L}$  where  $a_{s_i} \in \bar{a}$  and  $\varepsilon_i \in \{\pm 1\}$ . Bob also chooses his private key  $B = b_{t_1}^{\delta_1} \dots b_{t_L}^{\delta_L}$  where  $b_{t_i} \in \bar{b}$  and  $\delta_i \in \{\pm 1\}$ . Alice computes  $b'_i = A^{-1}b_iA$  for all  $b_i \in \bar{b}$  and sends it to Bob. Bob also computes  $a'_i = B^{-1}a_iB$  for all  $a_i \in \bar{a}$  and sends it to Alice. Now the shared secret key is  $K = A^{-1}B^{-1}AB$ . Alice can compute this key by

$$\begin{aligned} K_A &= A^{-1}a_{s_1}'^{\varepsilon_1} \dots a_{s_L}'^{\varepsilon_L} = A^{-1}(B^{-1}a_{s_1}B)^{\varepsilon_1} \dots (B^{-1}a_{s_L}B)^{\varepsilon_L} \\ &= A^{-1}B^{-1}a_{s_1}^{\varepsilon_1} \dots a_{s_L}^{\varepsilon_L}B = A^{-1}B^{-1}AB = K \end{aligned}$$

Bob can likewise compute  $K_B = B^{-1}b_{t_1}'^{\delta_1} \dots b_{t_L}'^{\delta_L} = B^{-1}A^{-1}BA$ , hence the shared key is  $K = K_B^{-1}$ .

In order to find  $K$ , the eaves-dropper needs to find either  $A' \in \langle a_1, \dots, a_{N_1} \rangle$  such that  $\bar{b}' = A'^{-1}\bar{b}A'$  or find  $B' \in \langle b_1, \dots, b_{N_2} \rangle$  such that  $\bar{a}' = B'^{-1}\bar{a}B'$ . Thus, the security of AAG is based on the assumption that the subgroup-restricted simultaneous conjugacy search problem is hard.

## 2. POLYCYCLIC GROUPS

In this section, we give a short review of polycyclic groups and discuss how we generate Heisenberg groups.

**2.1. Polycyclic groups.** Recall that  $G$  is a *polycyclic group* if it has a polycyclic series, i.e., a subnormal series  $G = G_1 \triangleright G_2 \triangleright \dots \triangleright G_{n+1} = \{1\}$  with non-trivial cyclic factors. The polycyclic generating sequence of  $G$  is the  $n$ -tuple  $(g_1, \dots, g_n)$  such that  $G_i = \langle g_i, G_{i+1} \rangle$  for  $1 \leq i \leq n$ .

Every polycyclic group has a finite presentation of the form:

$$\langle g_1, \dots, g_n \mid g_j^{g_i} = w_{ij}, g_j^{g_i^{-1}} = v_{ij}, g_k^{r_k} = u_{kk} \text{ for } 1 \leq i < j \leq n \text{ and } k \in I \rangle$$

where  $w_{ij}, v_{ij}, u_{kk}$  are words in the generators  $g_{i+1}, \dots, g_n$  and  $I$  is the set of  $i \in \{1, \dots, n\}$  such that  $r_i = [G_i : G_{i+1}]$  is finite. Here  $a^b$  stands for  $b^{-1}ab$ .

Using induction, we see that each element of  $G$  defined by this presentation can be uniquely written as  $g = g_1^{e_1} \dots g_n^{e_n}$  with  $e_i \in \mathbb{Z}$  for  $1 \leq i \leq n$ , and  $0 \leq e_i < r_i$  for  $i \in I$ . This is the *normal form* of an element. If every element in the group can be presented uniquely in normal form, then the polycyclic presentation is *consistent*. Note that every polycyclic group has a consistent polycyclic presentation. Thanks to the existence of normal form, the word problem in polycyclic groups can be solved efficiently, this is an important requirement of a platform group for AAG.

The *Hirsch length* of a polycyclic group is the number of  $i$  such that  $r_i = [G_i : G_{i+1}]$  is infinite. This number is invariant of the chosen polycyclic sequence.

For more details regarding polycyclic groups, see [13, 6].

**2.2. Heisenberg groups.** Heisenberg groups have been studied widely from the point of view of analysis, geometry, physics, etc [3]. From the group theory point of view, they are often used as examples of nilpotent groups. For more discussion of Heisenberg groups and their group-theoretic properties, see [7].

The three dimensional Heisenberg group, often known as *the* Heisenberg group, is the group of  $3 \times 3$  upper triangular matrices of the form

$$\begin{pmatrix} 1 & x & y \\ 0 & 1 & z \\ 0 & 0 & 1 \end{pmatrix}$$

where  $x, y, z \in \mathbb{R}$ . Another presentation for it is  $\langle a, b, c \mid [a, b] = c, [a, c] = [b, c] = 1 \rangle$ . The nilpotency for the Heisenberg group is easy to see, making it a polycyclic group.

Generalizing the Heisenberg group, we have higher dimension Heisenberg groups,  $H^{2n+1}$ ,  $n \geq 1, n \in \mathbb{Z}$ . As matrix group, they are groups of dimension  $n + 2$  matrices of the form

$$\begin{pmatrix} 1 & x_1 & \dots & x_n & c \\ 0 & 1 & 0 & \dots & y_n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & y_1 \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix}$$

where  $x_i, y_i, z \in \mathbb{R}$ . The calculation for the commutator subgroup and the center of  $H^{2n+1}$  is straight-forward, showing that it is also a nilpotent group, thus, also polycyclic. The Hirsch length of  $H^{2n+1}$  is  $2n + 1$ .

The Heisenberg groups of higher dimension  $H^{2n+1}$  also has presentation

$$\langle a_1, \dots, a_n, b_1, \dots, b_n, c \mid [a_i, b_i] = c, [a_i, c] = [b_i, c] = 1, [a_i, a_j] = [b_i, b_j] = 1, i \neq j \rangle$$

This presentation makes it easy to encode  $H^{2n+1}$  in a computer system. In fact, the GAP (Groups, Algorithms, Programming) system [8] can compute  $H^{2n+1}$  as part of the polycyclic package by Eick and Nickel [5]. Using this implementation of  $H^{2n+1}$ , we study the Heisenberg groups as platform group for the AAG key-exchange protocol, in particular, under the length-based attack, one of the major attacks for AAG.

### 3. LENGTH BASED ATTACK

**3.1. Overview of the length-based attack.** The length-based attack is a probabilistic attack against AAG, with the goal of finding Alice's (or Bob's) private key. It is based on the idea that a conjugation of the right element will decrease the length of the captured package. Using notation of Section 1, the captured package is  $\bar{b}' = (b'_1, \dots, b'_{N_2})$  where  $b'_i = A^{-1}b_iA$ . If we conjugate  $\bar{b}'$  with elements from the group  $\langle a_1, \dots, a_{N_1} \rangle$  and the resulting tuple has decreased length, then we know we have found a conjugating factor. The process of conjugation is then repeated with the decreased length tuple until another conjugating factor is found. The process ends when the conjugated captured package is the same as  $\bar{b} = (b_1, \dots, b_{N_2})$ , which is public knowledge. Then the conjugate can be recovered by reversing the sequence

of conjugating factors. The idea of the length-based attack can be summarized as going from bottom to top of the tower:

$$\begin{array}{c}
 b_i \\
 \downarrow \\
 a_{s_1}^{-\varepsilon_1} b_i a_{s_1}^{\varepsilon_1} \\
 \downarrow \\
 a_{s_2}^{-\varepsilon_2} a_{s_1}^{-\varepsilon_1} b_i a_{s_1}^{\varepsilon_1} a_{s_2}^{\varepsilon_2} \\
 \downarrow \\
 \vdots \\
 \downarrow \\
 a_{s_L}^{-\varepsilon_L} \dots a_{s_2}^{-\varepsilon_2} a_{s_1}^{-\varepsilon_1} b_i a_{s_1}^{\varepsilon_1} a_{s_2}^{\varepsilon_2} \dots a_{s_L}^{\varepsilon_L}
 \end{array}$$

For more details on the length-based attack, see [19, 20].

**3.2. LBA with memory 2.** In [11, 10, 20, 22, 9], several variations of the length-based attack are given. Here we recall Algorithm 4 of [9] which we use in our experiments.

In this algorithm,  $S$  holds  $M$  tuples every round and all elements of  $S$  are conjugated, but only the  $M$  smallest conjugated tuples (by total length) are added back into  $S$ . Because we are adding back  $M$  smallest tuples and not just one single element, we avoid the problem of the same element being removed and added time and again. Moreover, this method of saving several tuples, not just the ones whose length decreased after conjugation, keeps us away from problems generated by peaks. Here, for the stopping condition, we use a time-out that is defined by the user.

---

**Algorithm 1** LBA with Memory 2

---

- 1: Initialize  $S = \{(|\bar{b}'|, \bar{b}', id_G)\}$ .
  - 2: **while** not time out **do**
  - 3:   **for**  $(|\bar{c}|, \bar{c}, x) \in S$  **do**
  - 4:     Remove  $(|\bar{c}|, \bar{c}, x)$  from  $S$
  - 5:     Compute  $\bar{c}^{a_i^\varepsilon}$  for all  $i \in \{1 \dots N_1\}$  and  $\varepsilon = \pm 1$
  - 6:     **if**  $\bar{c}^{a_i^\varepsilon} = \bar{b}$  **then** output inverse of  $xa_i^\varepsilon$  and stop
  - 7:     Save  $(|\bar{c}^{a_i^\varepsilon}|, \bar{c}^{a_i^\varepsilon}, xa_i^\varepsilon)$  in  $S'$
  - 8:   **end for**
  - 9:   After finished all conjugations, sort  $S'$  by the first element of every tuple
  - 10:   Copy the smallest  $M$  elements into  $S$  and delete the rest of  $S'$
  - 11: **end while**
  - 12: Otherwise, output FAIL
- 

#### 4. RESULTS

As a test of the resilience of polycyclic groups against the length-based attack, Garber, Kahrobaei, and Lam [9] implemented four variants of the length-based attack and performed experiments against all four variants. The conclusion was that Algorithm 4, LBA with Memory 2, was the most effective. Hence, in this study, we use Algorithm 4 for all of our experiments.

We performed several sets of tests, all of which were run on an Intel Core I7 quad-core 2.0GHz computer with 12GB of RAM, running Ubuntu version 12.04 with GAP version 4.5 and 10GB of memory allowance. In all these tests, the Heisenberg group  $G$  with  $2n$  generators having Hirsch length  $h(G) = 2n + 1$  is generated using GAP with the Polycyclic group package [5]. The size of Alices and Bobs public sets are both  $N_1 = N_2 = 20$ , the memory used is  $M = 1000$  and the time-out is 30 minutes.

To see the effect of element length, we fix the key length  $L = 10$ , but changes the range of random element between  $[L_1, L_2] = [10, 13]$  and  $[20, 23]$ . The result is as follows:

n	h(G)	$[L_1, L_2] = [10, 13]$	$[L_1, L_2] = [20, 23]$
5	11	29%	53%
6	13	69%	39%
7	15	51%	58%
8	17	62%	67%

As we can see, changing the length of random element does not have a dramatic effect on the success rate. Hence, to ensure a lower rate of success, one should look for other factors like the Hirsch length or the key length.

To see the effect of Hirsch length, we use a small key length  $L = 10$  to increase the possibility of success. The length of each random element is in  $[L_1, L_2] = [10, 13]$ . The result is as follows:

n	h(G)	Time	Success rate
3	7	45.42 hours	11%
5	11	37.82 hours	29%
6	13	21.12 hours	69%
7	15	28.95 hours	51%
8	17	24.33 hours	62%

Curiously, the higher the Hirsch length in this case, the higher and quicker the length-based attack succeeded. This interesting result warrant further investigations.

To illustrate the effect of key length, in the following experiment, we let the length of each random element to be in  $[L_1, L_2] = [20, 23]$  and vary the key length  $L$ .

n	h(G)	L=10	L=20	L=50
5	11	53%	11%	1%
6	13	39%	7%	0%
7	15	58%	5%	1%
8	17	67%	9%	7%

Clearly, increasing the key length reduced the success rate dramatically. What is interesting is that the same algorithm, together with the same parameters, when applied on polycyclic group of Hirsch length 10 generated with number field as in [9] gave 0% success rate. However, it is clear that we need different parameters to ensure low success rates.

We turn the key length to  $L = 50$ , let length of random element be  $[L_1, L_2] = [40, 43]$ . Note that this is the parameters used in [20], which succeeded in breaking AAG with braid group.

n	$h(G)$	$L=50$
6	13	0%
7	15	0%
8	17	1%

As expected, this produces almost zero success rate. We recommend this as parameters for Heisenberg group as platform for AAG.

Finally, as a stress test, we increase the time-out to 4 hours instead of the usual 30 minutes per test. Because of the long time-out, we only do 50 tests each, instead of 100 test batch as in the previous experiments. The element length is kept at  $[L_1, L_2] = [20, 23]$ , key length is  $L = 20$ .

n	$h(G)$	$L=20$
6	13	3%
7	15	4%
8	17	7%

Even at such long time-out, success rate is quite modest, given that the key length we use here is quite small.

With these results, we conclude that the Heisenberg groups work well as platform for the AAG protocol given the correct parameters. This strengthens the idea to use polycyclic group as platform for AAG in particular, and non-commutative cryptographic primitives in general.

## REFERENCES

- [1] I. Anshel, M. Anshel, and D. Goldfeld. An algebraic method for public-key cryptography. *Mathematical Research Letters*, 6:287–291, 1999.
- [2] M. Anshel and D. Kahrobaei. Decision and search in non-abelian Cramer-Shoup public key cryptosystem. *Groups, Complexity, Cryptology*, 1:217–225, 2009.
- [3] Ernst Binz and Sonja Pods. *The geometry of Heisenberg groups: with applications in signal theory, optics, quantization, and field quantization*, volume 151. American Mathematical Soc., 2008.
- [4] B. Eick and D. Kahrobaei. Polycyclic groups: a new platform for cryptography? Technical report, 2004. <http://arxiv.org/abs/math/0411077>.
- [5] B. Eick and W. Nickel. *Polycyclic: Computation with polycyclic groups, a GAP 4 package*, <http://www.gap-system.org/Packages/polycyclic.html>.
- [6] Bettina Eick. Algorithms for polycyclic groups. Habilitationsschrift, Universitt Kassel, 2000, 113 p.
- [7] David Epstein, MS Paterson, JW Cannon, DF Holt, SV Levy, and William P Thurston. *Word processing in groups*. AK Peters, Ltd., 1992.
- [8] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.7.2*, <http://www.gap-system.org>, 2013.
- [9] D. Garber, D. Kahrobaei, and H. T. Lam. Analyzing the Length-Based Attack on Polycyclic Groups. May 2013. <http://arxiv.org/abs/1305.0548>.
- [10] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, and U. Vishne. Probabilistic solutions of equations in the braid group. *Advances in Applied Mathematics* 35, pages 323–334, 2005.
- [11] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, and U. Vishne. Length-based conjugacy search in the braid group. *Contemporary Mathematics* 418, pages 75–87, 2006.
- [12] Dennis Hofheinz and Rainer Steinwandt. *A practical attack on some braid group based cryptographic primitives*. Springer, 2002.
- [13] D. F. Holt, B. Eick, and E. A. O’Brien. *Handbook of computational group theory*. Chapman & Hall CRC, 2005.
- [14] J. Hughes and A. Tannenbaum. Length-based attacks for certain group based encryption rewriting systems. *Workshop SECI02 Securite de la Communication sur Internet*, 2002.

- [15] D. Kahrobaei and B. Khan. A non-commutative generalization of El-Gamal key exchange using polycyclic groups. *Proceedings of the Global Telecommunications Conference*, 4(2), 2006.
- [16] D. Kahrobaei and C. Koupparis. Non-commutative digital signatures using non-commutative groups. *Groups, Complexity, Cryptology*, 4, 2012.
- [17] K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J. Kang, and C. Park. New public-key cryptosystem using braid groups. *Advances in cryptology CRYPTO 2000 (Santa Barbara, CA)*, LNCS, 1880:166–183, 2000.
- [18] Sang Jin Lee and Eonkyung Lee. Potential weaknesses of the commutator key agreement protocol based on braid groups. In *Advances in CryptologyEUROCRYPT 2002*, pages 14–28. Springer, 2002.
- [19] A. Myasnikov, V. Shpilrain, and A. Ushakov. *Non-commutative Cryptography and Complexity of Group-theoretic Problems*. American Mathematical Society, 2011.
- [20] A. D. Myasnikov and A. Ushakov. Length based attack and braid groups: Cryptanalysis of Anshel-Anshel-Goldfeld key exchange protocol. *PKC 2007, LNCS 4450*, pages 76–88, 2007.
- [21] Alexei Myasnikov, Vladimir Shpilrain, and Alexander Ushakov. Random subgroups of braid groups: an approach to cryptanalysis of a braid group based cryptographic protocol. In *Public Key Cryptography-PKC 2006*, pages 302–314. Springer, 2006.
- [22] D. Ruinskiy, A. Shamir, and B. Tsaban. Length-based cryptanalysis: the case of Thompsons group. *Journal of Mathematical Cryptology* 1, pages 359–372, 2007.
- [23] V. Shpilrain and A. Ushakov. Thompson’s group and public key cryptography. *ACNS’05 Proceedings of the Third international conference on Applied Cryptography and Network Security*, 2005.